Report on Multiple-Choice Quiz App with Timer and Score Tracking using JavaScript and React

---

Team Members – Group 8

Roll No	Name
21	Nideesh Kayal
22	Yohaan Khan Pathan
49	Devansh Singh
50	Ganesh Singh

---

1. Abstract

The project titled "Multiple-Choice Quiz App with Timer and Score Tracking using JavaScript and React" presents an innovative approach to digital learning through interactive quizzes. This web-based application leverages modern front-end technologies to deliver an engaging and educational user experience. Key features include multiple-choice questions, a dynamic countdown timer, score tracking, and immediate feedback for users. Built using the React framework and styled with Tailwind CSS, this app showcases how component-based architecture and state management in React can be effectively utilized to create real-world educational tools. This report outlines the motivation, architecture, implementation, and outcomes of this application.

2. Introduction

Quizzes have always been a popular tool in education and training for evaluating knowledge and reinforcing learning. With the rapid adoption of digital learning platforms, quiz applications have evolved from static forms to dynamic, user-centric interfaces. The aim of our project was to design and develop a responsive quiz application that incorporates a real-time countdown timer, score tracking system, and feedback mechanism.

By using React—a leading JavaScript library for building user interfaces—we created a quiz system that is modular, scalable, and maintainable. The app is built to function seamlessly in a web browser and offers features like instant score calculation and a visually appealing design using Tailwind CSS. This project not only served as a practical demonstration of our understanding of React and JavaScript but also reflected how such technologies can be used in educational contexts.

3. Background

3.1 React and JavaScript React is an open-source JavaScript library maintained by Meta. It facilitates the development of complex user interfaces through reusable components and

unidirectional data flow. React makes it easier to build interactive UIs that respond efficiently to user inputs and state changes.

3.2 Tailwind CSS Tailwind is a utility-first CSS framework that enables developers to rapidly build custom user interfaces. In this project, Tailwind played a significant role in styling the quiz cards, buttons, and timer displays.

3.3 Lucide React & Framer Motion Lucide React icons added visual clarity and aesthetics to various UI elements like the timer. Framer Motion was used to animate transitions between quiz questions and score displays, enhancing user engagement.

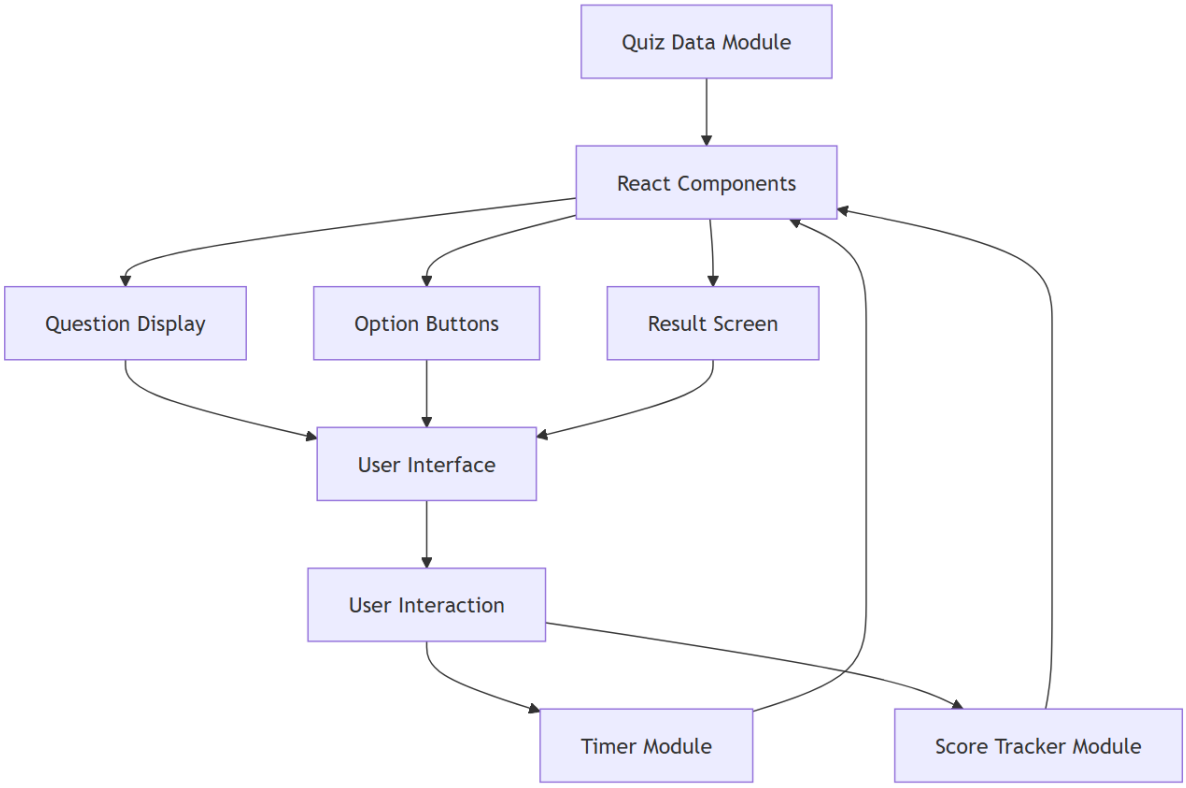
3.4 Need for Timer and Score Tracking Adding a timer increases the challenge and simulates a real-exam scenario. Score tracking provides feedback and motivation, allowing users to assess their knowledge and improve over time.

4. System Description

4.1 Architecture Overview The application follows a component-based design pattern:

- **App.tsx**: Root component managing quiz flow and state (current question, score, time).
- **QuizCard.tsx**: Displays the current question, options, and handles answer logic.
- **Timer.tsx**: Presents the countdown timer componentally.

4.2 Module Diagram:



4.3 Component Responsibilities:

- **App.tsx**: Maintains state using React's useState and useEffect hooks. Manages current question index, score, and timer countdown.
- **QuizCard.tsx**: Accepts props such as the question object and a callback function. Displays the question, maps answer options to buttons, and handles user selection.
- **Timer.tsx**: Calculates remaining minutes and seconds from total time and displays in "MM:SS" format. Uses Lucide icons for visual enhancement.

4.4 Code Explanation

App.tsx (Excerpt)

```
const [currentQuestion, setCurrentQuestion] = useState(0);
const [score, setScore] = useState(0);
const [timeRemaining, setTimeRemaining] = useState(300);
```

- Initializes state for question index, score, and time.

```
useEffect(() => {
  if (timeRemaining <= 0) return;
  const timer = setInterval(() => setTimeRemaining(prev => prev - 1), 1000);
  return () => clearInterval(timer);
}, [timeRemaining]);
```

- Implements countdown using setInterval inside useEffect. Clears the interval on unmount.

QuizCard.tsx (Excerpt)

```
const handleAnswer = (option: string) => {
  if (option === question.correctAnswer) onCorrect();
  setSelected(option);
  setTimeout(onNext, 1000);
};
```

- Handles logic when an answer is clicked. If correct, calls onCorrect() to increment score, then moves to next question.

Timer.tsx

```
const minutes = Math.floor(timeRemaining / 60);
```

const seconds = timeRemaining % 60;

- Converts total seconds to a readable minute:second format.

5. Results and Discussions

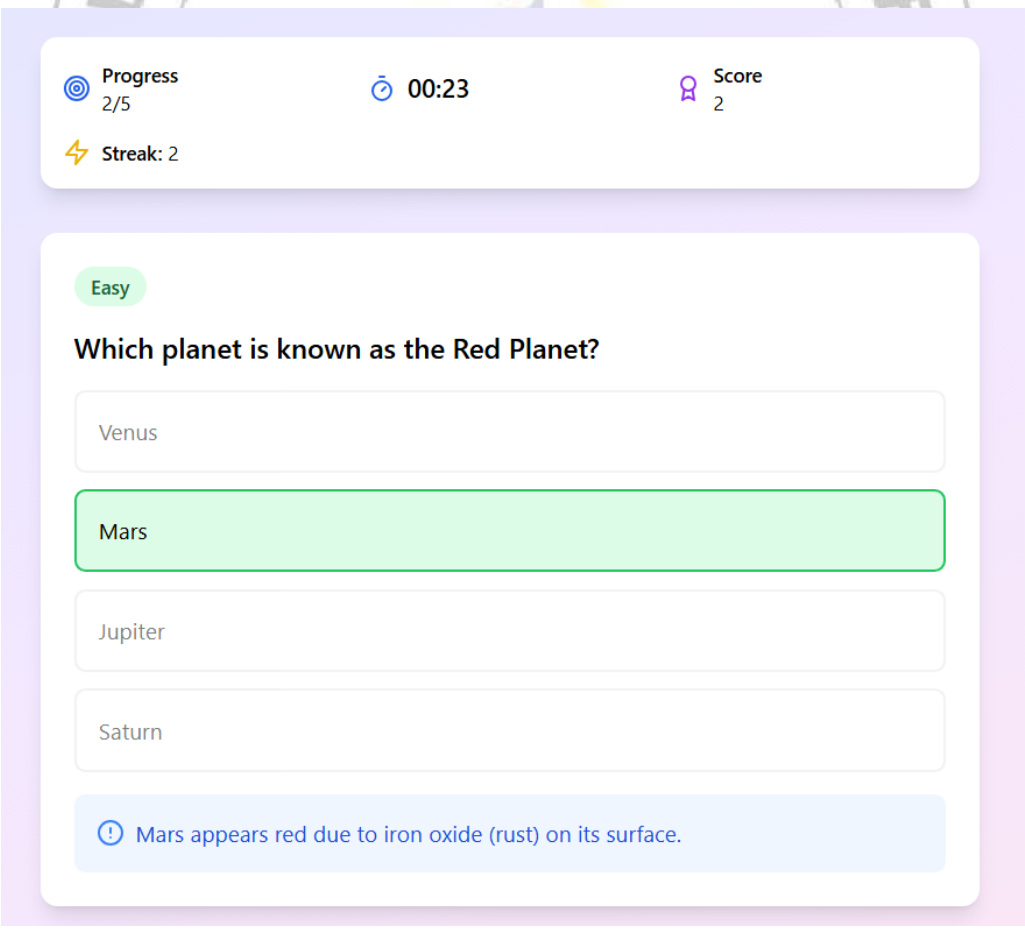
The app was tested with different quiz datasets and showed accurate performance in terms of score calculation, timer decrement, and responsive design.

Features Successfully Implemented:

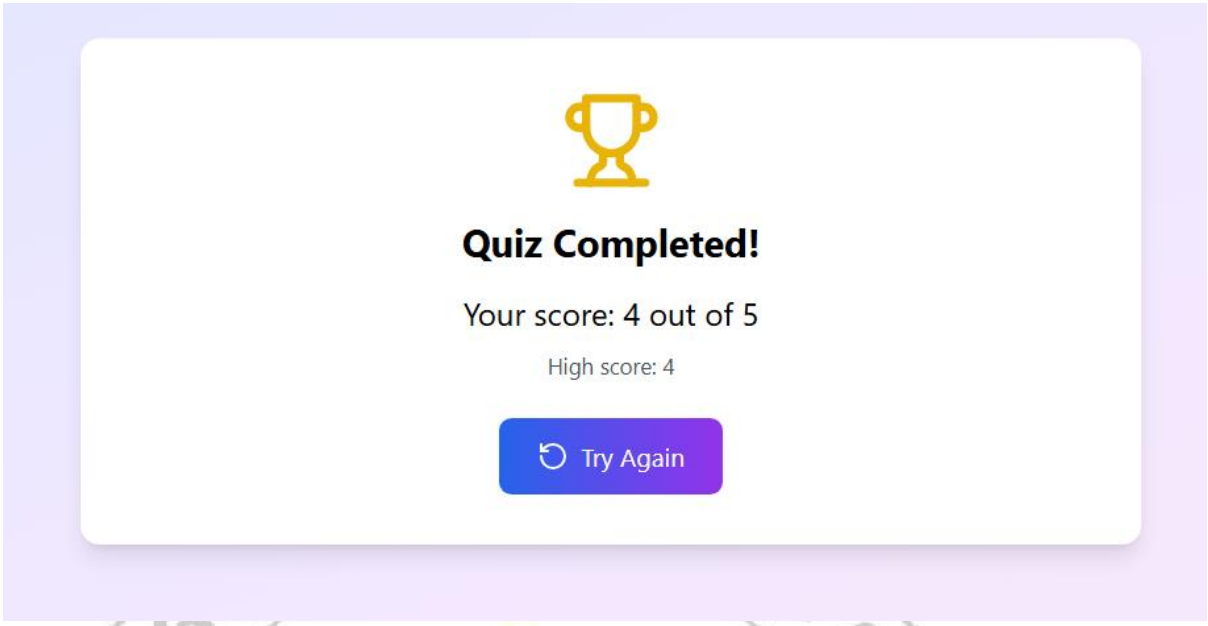
- Dynamic rendering of questions
- Real-time score updates
- Countdown timer with visual indication
- Immediate answer feedback
- Clean and intuitive user interface

Below are a few screenshots from the final application:

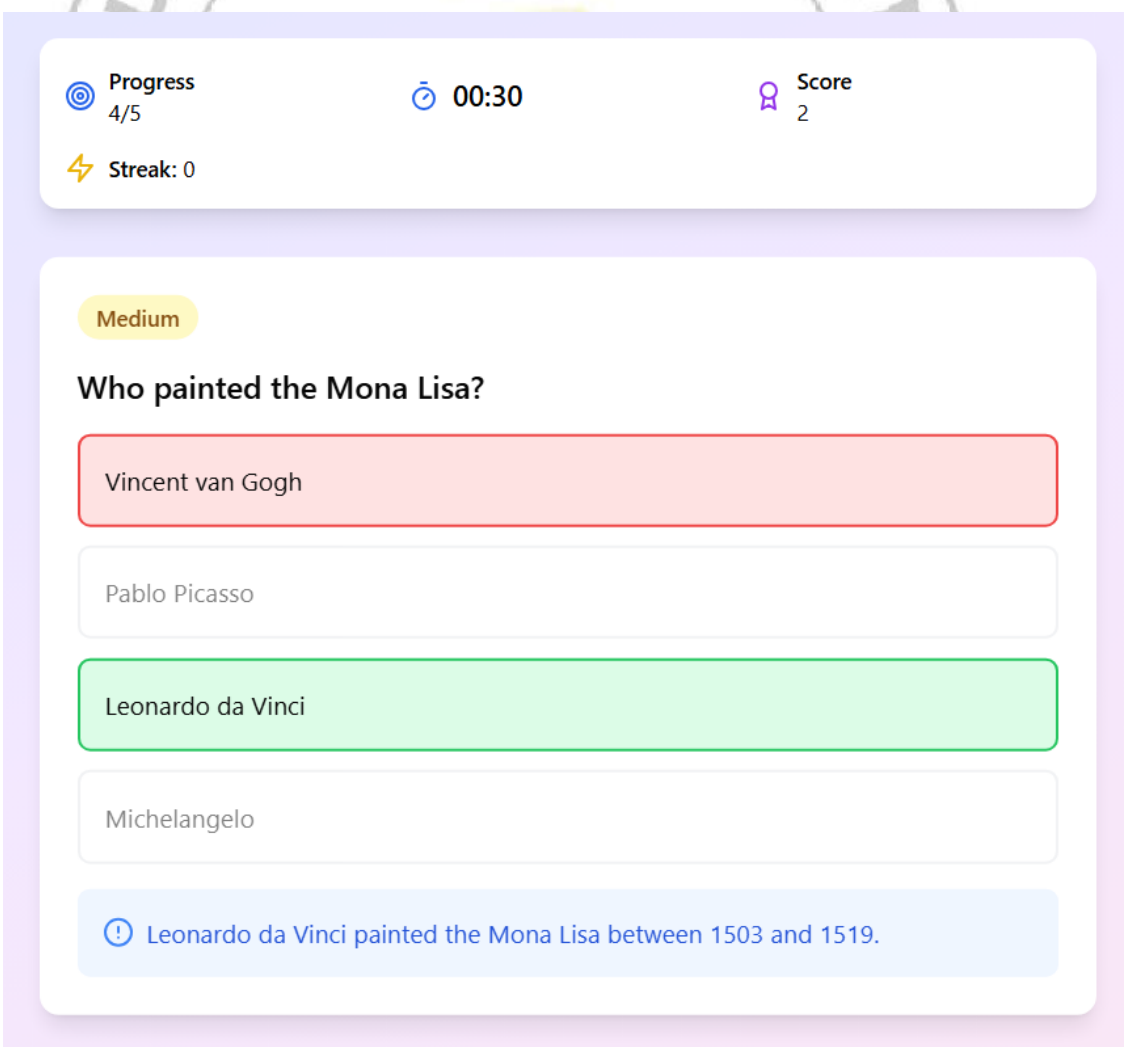
1. Interface on correct answer



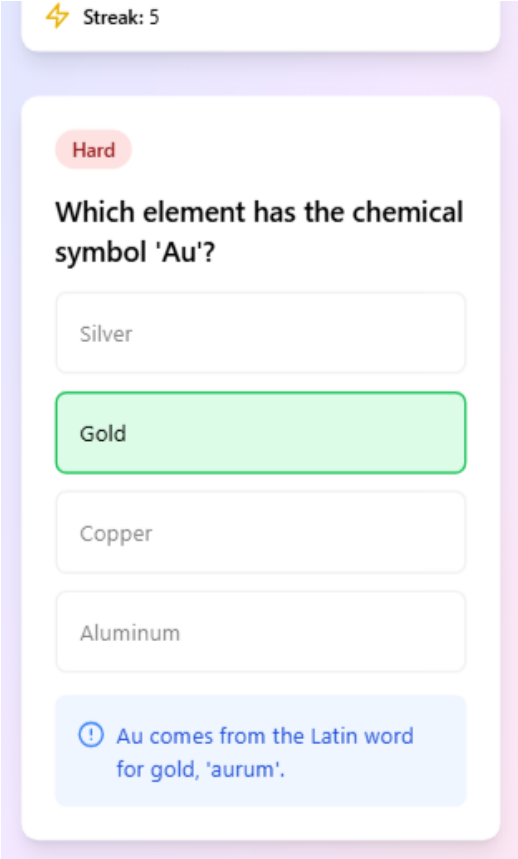
2. Result page



3. Interface on incorrect answer



4. Mobile Interface emulation



User Feedback:

- Appreciated minimal UI and clear layout
- Found the timer motivating and challenging
- Suggested adding question difficulty levels and topics

6. Conclusion and Future Scope

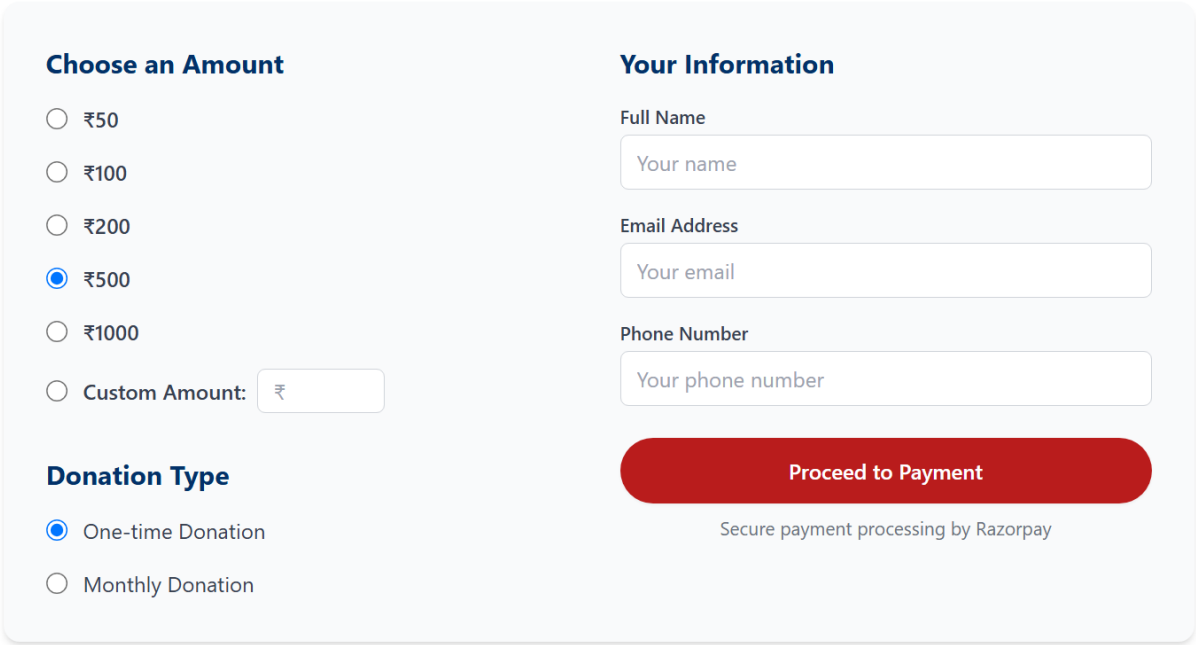
The "Multiple-Choice Quiz App" successfully fulfills the goal of building a modular, interactive, and educational quiz application using React. The app demonstrates effective state management, UI composition, and user engagement strategies.

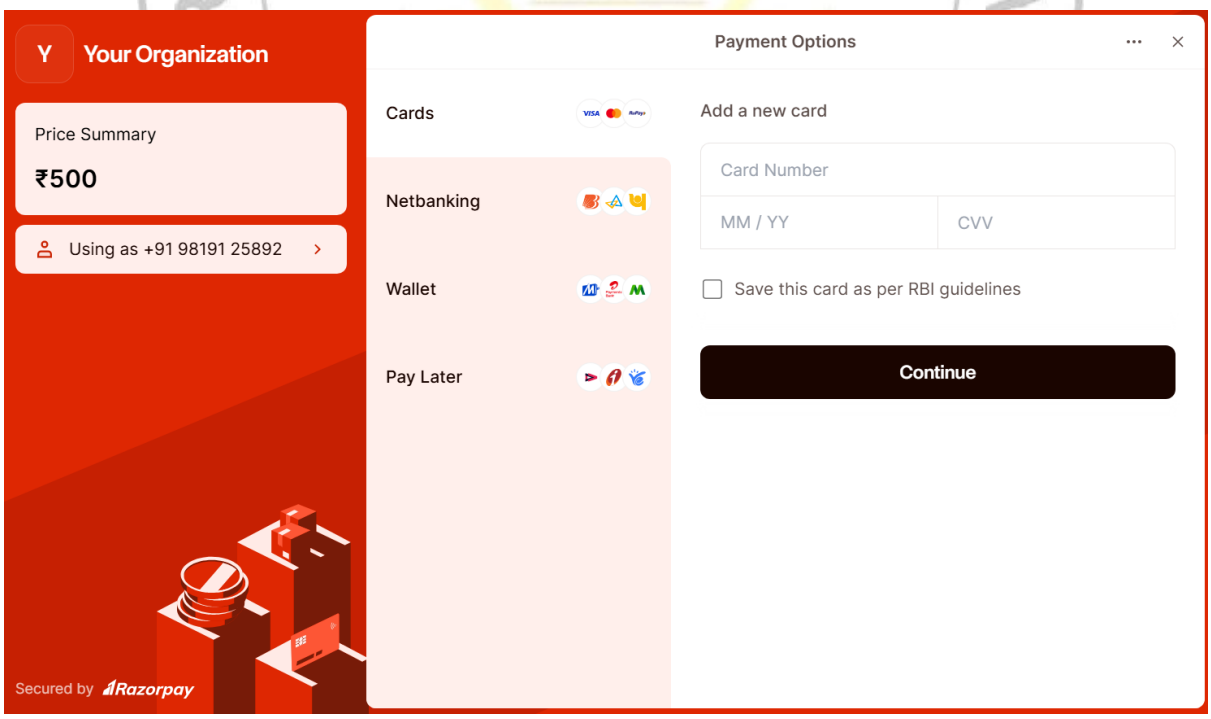
Future Enhancements:

- Support for random question generation from API
- User login and history tracking
- Leaderboards and quiz analytics
- Admin panel for quiz management

- Dark mode and accessibility enhancements
- Option for timed vs untimed mode

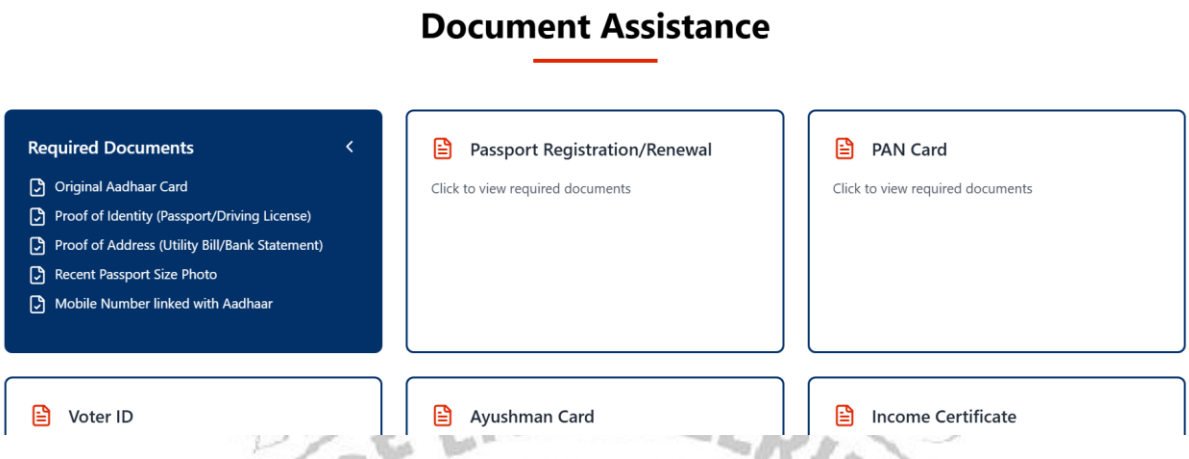
This project also served as a basis for a complete website, consisting of a gallery, functional pages, servers, backend along with payment integration API, for an NGO. Here are a few screenshots of the payment API integration:



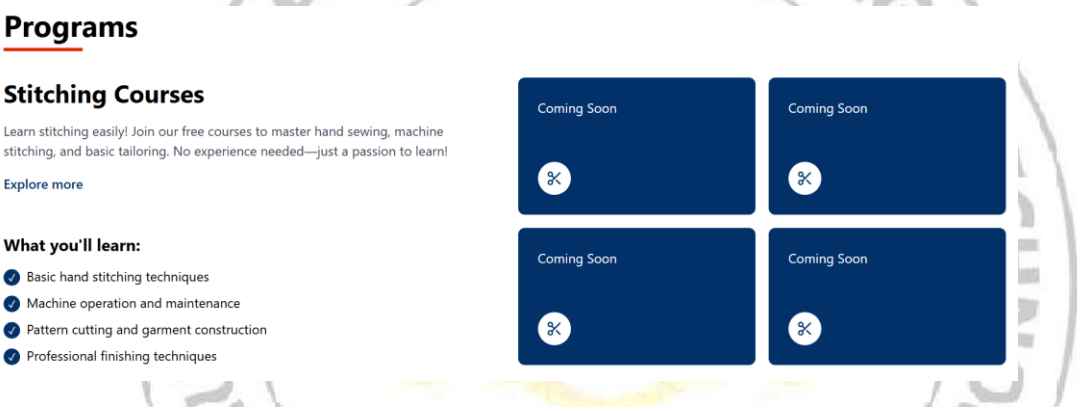


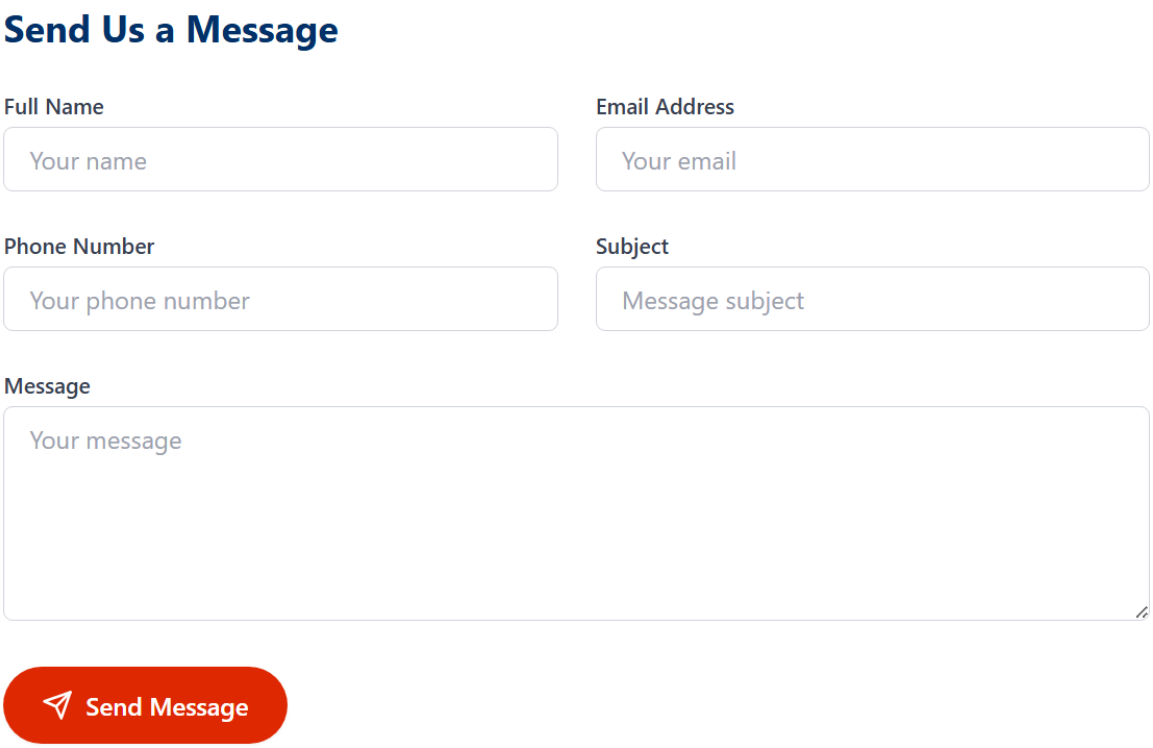
This payment module can also be used for monetization of premium features for the quiz app.

The same website also has a document assistance page, which has flappable cards made using React which can also be integrated into the quiz app as flashcards for educational content.



Here are a few other layouts that can be used in the quiz app:





7. References

- React Documentation: https://reactjs.org/
- Tailwind CSS: https://tailwindcss.com/
- Framer Motion Docs: https://www.framer.com/motion/
- Lucide Icons: https://lucide.dev/
- Stack Overflow Community
- GitHub Repositories for Quiz Apps (inspiration)

8. Annexure (Code)

App.tsx

```
import React, { useState, useEffect } from 'react';
import { motion, AnimatePresence } from 'framer-motion';
import { Timer } from './components/Timer';
import { QuizCard } from './components/QuizCard';
import { questions } from './data/questions';
import { QuizState } from './types';
import { Trophy, RotateCcw, Zap, Target, Award } from 'lucide-react';

const INITIAL_TIME = 30; // seconds per question

function App() {
  const [state, setState] = useState<QuizState>({
    currentQuestionIndex: 0,
    score: 0,
    timeRemaining: INITIAL_TIME,
    selectedAnswer: null,
    isAnswered: false,
    isFinished: false,
    difficulty: 'all',
    streak: 0,
    highScore: 0
  });

  useEffect(() => {
    const savedHighScore = localStorage.getItem('quizHighScore');
    if (savedHighScore) {
      setState(prev => ({ ...prev, highScore: parseInt(savedHighScore, 10) }));
    }
```

```
}, []);

useEffect(() => {
  if (state.timeRemaining > 0 && !state.isAnswered && !state.isFinished) {
    const timer = setInterval(() => {
      setState(prev => ({
        ...prev,
        timeRemaining: prev.timeRemaining - 1,
      }));
    }, 1000);

    return () => clearInterval(timer);
  } else if (state.timeRemaining === 0 && !state.isAnswered) {
    handleAnswer(-1);
  }
}, [state.timeRemaining, state.isAnswered]);

const handleAnswer = (selectedIndex: number) => {
  const currentQuestion = questions[state.currentQuestionIndex];
  const isCorrect = selectedIndex === currentQuestion.correctAnswer;

  const newScore = isCorrect ? state.score + 1 : state.score;
  const newStreak = isCorrect ? state.streak + 1 : 0;
  const newHighScore = Math.max(state.highScore, newScore);

  if (newHighScore > state.highScore) {
    localStorage.setItem('quizHighScore', newHighScore.toString());
  }

  setState(prev => ({
    ...prev,
    score: newScore,
    streak: newStreak,
    highScore: newHighScore,
    selectedAnswer: selectedIndex,
    isAnswered: true,
  }));

  setTimeout(() => {
    if (state.currentQuestionIndex < questions.length - 1) {
      setState(prev => ({
        ...prev,
```

```
        currentQuestionIndex: prev.currentQuestionIndex + 1,
        timeRemaining: INITIAL_TIME,
        selectedAnswer: null,
        isAnswered: false,
       }));
     } else {
       setState(prev => ({
        ...prev,
        isFinished: true,
       }));
     }
   }, 2000);
 };

 const resetQuiz = () => {
   setState({
     currentQuestionIndex: 0,
     score: 0,
     timeRemaining: INITIAL_TIME,
     selectedAnswer: null,
     isAnswered: false,
     isFinished: false,
     difficulty: 'all',
     streak: 0,
     highScore: state.highScore
   });
 };

 return (
   <div className="min-h-screen bg-gradient-to-br from-indigo-100 via-purple-100 to-pink-
100 flex flex-col items-center py-8 px-4">
     <div className="w-full max-w-2xl">
       {!state.isFinished ? (
         <>
           <motion.div
             initial={{ opacity: 0, y: -20 }}
             animate={{ opacity: 1, y: 0 }}
             className="bg-white rounded-xl shadow-lg p-4 mb-8"
           >
             <div className="grid grid-cols-3 gap-4">
               <div className="flex items-center gap-2">
                 <Target className="w-5 h-5 text-blue-600" />
```

```
          <div className="text-sm">
            <div className="font-semibold">Progress</div>
            <div>{state.currentQuestionIndex + 1}/{questions.length}</div>
          </div>
        </div>

        <div className="flex items-center gap-2">
          <Timer timeRemaining={state.timeRemaining} />
        </div>

        <div className="flex items-center gap-2">
          <Award className="w-5 h-5 text-purple-600" />
          <div className="text-sm">
            <div className="font-semibold">Score</div>
            <div>{state.score}</div>
          </div>
        </div>
      </div>

      <div className="mt-4 flex items-center gap-2">
        <Zap className="w-5 h-5 text-yellow-500" />
        <div className="text-sm">
          <span className="font-semibold">Streak:</span> {state.streak}
        </div>
      </div>
    </motion.div>

    <AnimatePresence mode="wait">
      <QuizCard
        key={state.currentQuestionIndex}
        question={questions[state.currentQuestionIndex]}
        selectedAnswer={state.selectedAnswer}
        isAnswered={state.isAnswered}
        onSelectAnswer={handleAnswer}
      />
    </AnimatePresence>
  </>
) : (
  <motion.div
    initial={{ opacity: 0, scale: 0.9 }}
    animate={{ opacity: 1, scale: 1 }}
    className="bg-white rounded-xl shadow-lg p-8 text-center"
```

```
        >
          <Trophy className="w-16 h-16 text-yellow-500 mx-auto mb-4" />
          <h2 className="text-2xl font-bold mb-4">Quiz Completed!</h2>
          <div className="space-y-2 mb-6">
            <p className="text-xl">
              Your score: {state.score} out of {questions.length}
            </p>
            <p className="text-sm text-gray-600">
              High score: {state.highScore}
            </p>
            {state.score === questions.length && (
              <motion.div
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                className="text-green-600 font-semibold"
              >
                Perfect Score! 🎉
              </motion.div>
            )}
          </div>
          <motion.button
            whileHover={{ scale: 1.05 }}
            whileTap={{ scale: 0.95 }}
            onClick={resetQuiz}
            className="flex items-center gap-2 mx-auto px-6 py-3 bg-gradient-to-r from-blue-600 to-purple-600 text-white rounded-lg hover:from-blue-700 hover:to-purple-700 transition-all duration-200"
          >
            <RotateCcw className="w-5 h-5" />
            Try Again
          </motion.button>
        </motion.div>
      )}
    </div>
  </div>
 );
}

export default App;
```

QuizCard.tsx

```
import React from 'react';
```

```
import { motion } from 'framer-motion';
import { Question } from '../types';
import { AlertCircle } from 'lucide-react';

interface QuizCardProps {
  question: Question;
  selectedAnswer: number | null;
  isAnswered: boolean;
  onSelectAnswer: (index: number) => void;
}

export const QuizCard: React.FC<QuizCardProps> = ({
  question,
  selectedAnswer,
  isAnswered,
  onSelectAnswer,
}) => {
  const getOptionClass = (index: number) => {
    if (!isAnswered) {
      return selectedAnswer === index ? 'bg-blue-100 border-blue-500' : 'hover:bg-gray-50';
    }

    if (index === question.correctAnswer) {
      return 'bg-green-100 border-green-500';
    }

    if (selectedAnswer === index) {
      return 'bg-red-100 border-red-500';
    }

    return 'opacity-50';
  };

  const getDifficultyColor = (difficulty: string) => {
    switch (difficulty) {
      case 'easy': return 'bg-green-100 text-green-800';
      case 'medium': return 'bg-yellow-100 text-yellow-800';
      case 'hard': return 'bg-red-100 text-red-800';
      default: return 'bg-gray-100 text-gray-800';
    }
  };
```

```
  return (
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      exit={{ opacity: 0, y: -20 }}
      className="w-full max-w-2xl bg-white rounded-xl shadow-lg p-6"
    >
      <div className="flex justify-between items-center mb-4">
        <span className={`px-3 py-1 rounded-full text-sm font-medium
${getDifficultyColor(question.difficulty)}`}>
          {question.difficulty.charAt(0).toUpperCase() + question.difficulty.slice(1)}
        </span>
      </div>

      <h2 className="text-xl font-semibold mb-4">{question.question}</h2>

      <div className="space-y-3">
        {question.options.map((option, index) => (
          <motion.button
            key={index}
            whileHover={{ scale: !isAnswered ? 1.01 : 1 }}
            whileTap={{ scale: !isAnswered ? 0.99 : 1 }}
            onClick={() => !isAnswered && onSelectAnswer(index)}
            disabled={isAnswered}
            className={`w-full p-4 text-left rounded-lg border-2 transition-all duration-200
              ${getOptionClass(index)}`}
          >
            {option}
          </motion.button>
        ))}
      </div>

      {isAnswered && question.explanation && (
        <motion.div
          initial={{ opacity: 0, height: 0 }}
          animate={{ opacity: 1, height: 'auto' }}
          className="mt-4 p-4 bg-blue-50 rounded-lg flex items-start gap-2"
        >
          <AlertCircle className="w-5 h-5 text-blue-500 flex-shrink-0 mt-0.5" />
          <p className="text-blue-700">{question.explanation}</p>
        </motion.div>
      )}
```

```
    </motion.div>
  );
};
```

Timer.tsx

```
import React from 'react';
import { Timer as TimerIcon } from 'lucide-react';

interface TimerProps {
  timeRemaining: number;
}

export const Timer: React.FC<TimerProps> = ({ timeRemaining }) => {
  const minutes = Math.floor(timeRemaining / 60);
  const seconds = timeRemaining % 60;

  return (
    <div className="flex items-center gap-2 text-lg font-semibold">
      <TimerIcon className="w-5 h-5 text-blue-600" />
      <span>
        {String(minutes).padStart(2, '0')}:{String(seconds).padStart(2, '0')}
      </span>
    </div>
  );
};
```